Interfaces

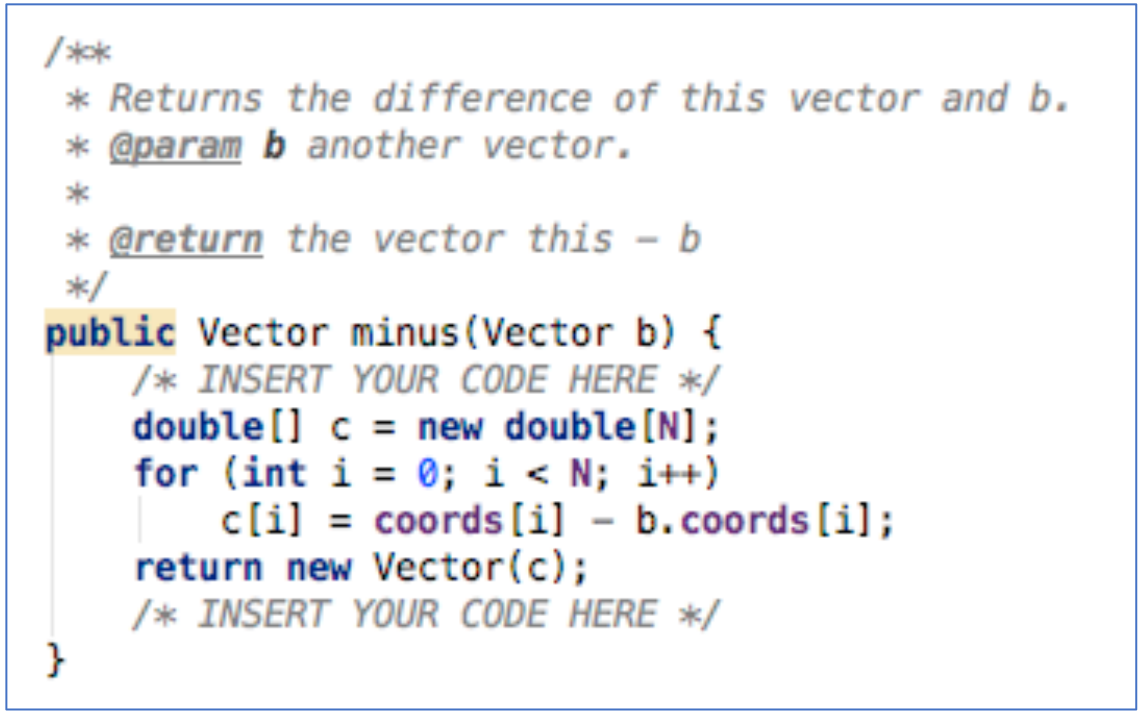
CSC 209 Data Structures

Kulwadee Somboonviwat

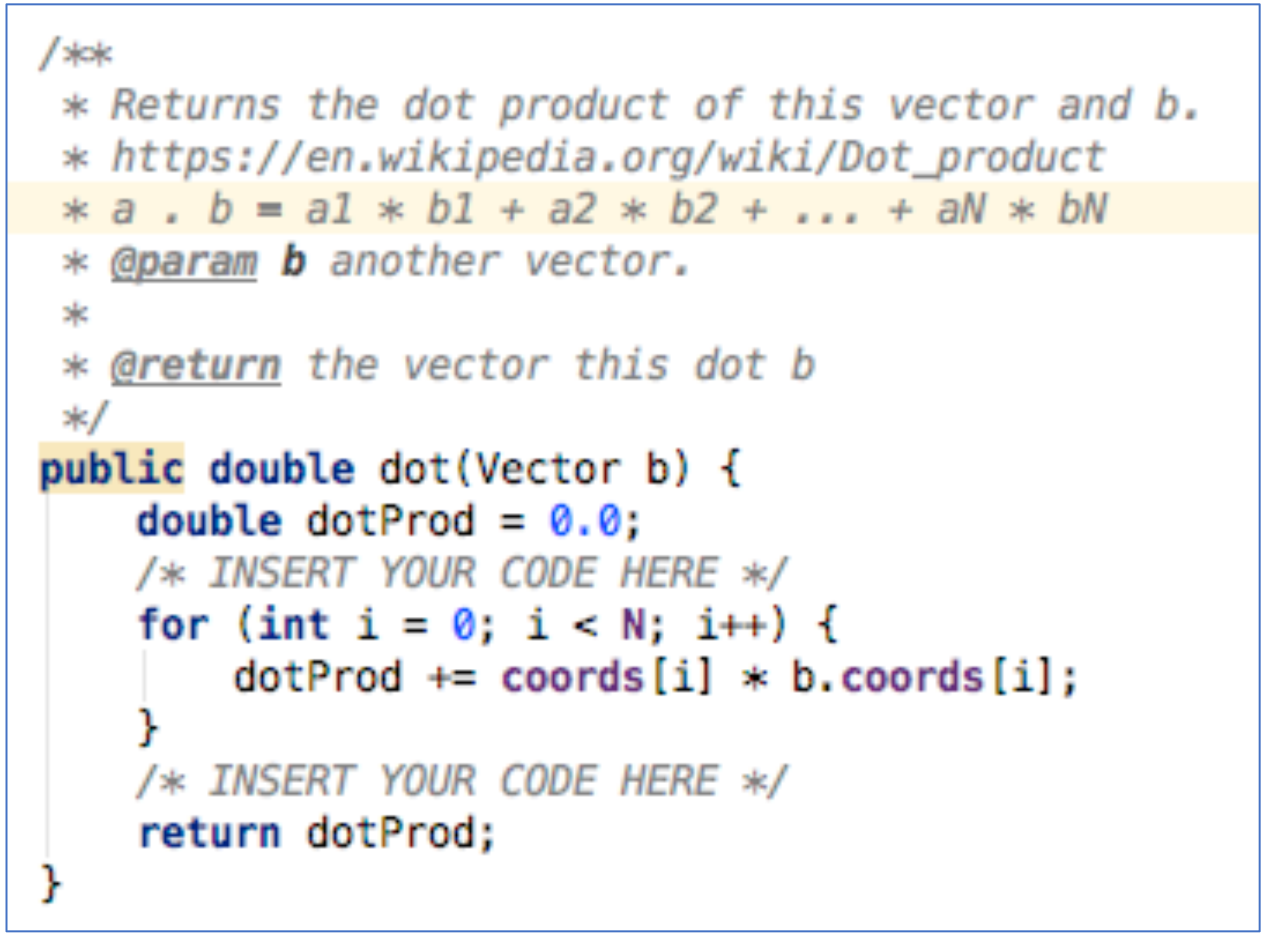
kulwadee.som [at] sit.kmutt.ac.th

Solution to Lecture 1’s Practice Exercise

Go to CSC209 GitHub page for source code: https://github.com/kulwadeesom/csc209_256002



```
/**
 * Returns the difference of this vector and b.
 * @param b another vector.
 *
 * @return the vector this - b
 */
public Vector minus(Vector b) {
    /* INSERT YOUR CODE HERE */
    double[] c = new double[N];
    for (int i = 0; i < N; i++)
        c[i] = coords[i] - b.coords[i];
    return new Vector(c);
    /* INSERT YOUR CODE HERE */
}
```



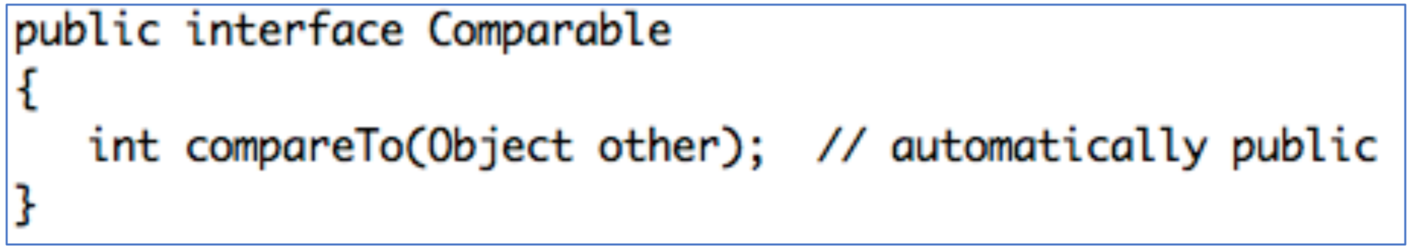
```
/**
 * Returns the dot product of this vector and b.
 * https://en.wikipedia.org/wiki/Dot_product
 * a . b = a1 * b1 + a2 * b2 + ... + aN * bN
 * @param b another vector.
 *
 * @return the vector this dot b
 */
public double dot(Vector b) {
    double dotProd = 0.0;
    /* INSERT YOUR CODE HERE */
    for (int i = 0; i < N; i++) {
        dotProd += coords[i] * b.coords[i];
    }
    /* INSERT YOUR CODE HERE */
    return dotProd;
}
```

Interfaces

- a set of requirements for the classes that want to conform to the interface.
- Conforming class must provide implementations of all methods specified by the interface.
- Interface is **not** a class (thus, cannot be instantiated with the **new** operator).
- A class can implement **multiple interfaces** (but can inherit from only one superclass).
- Use cases:
  - Service provider E.g. **Arrays.sort**:

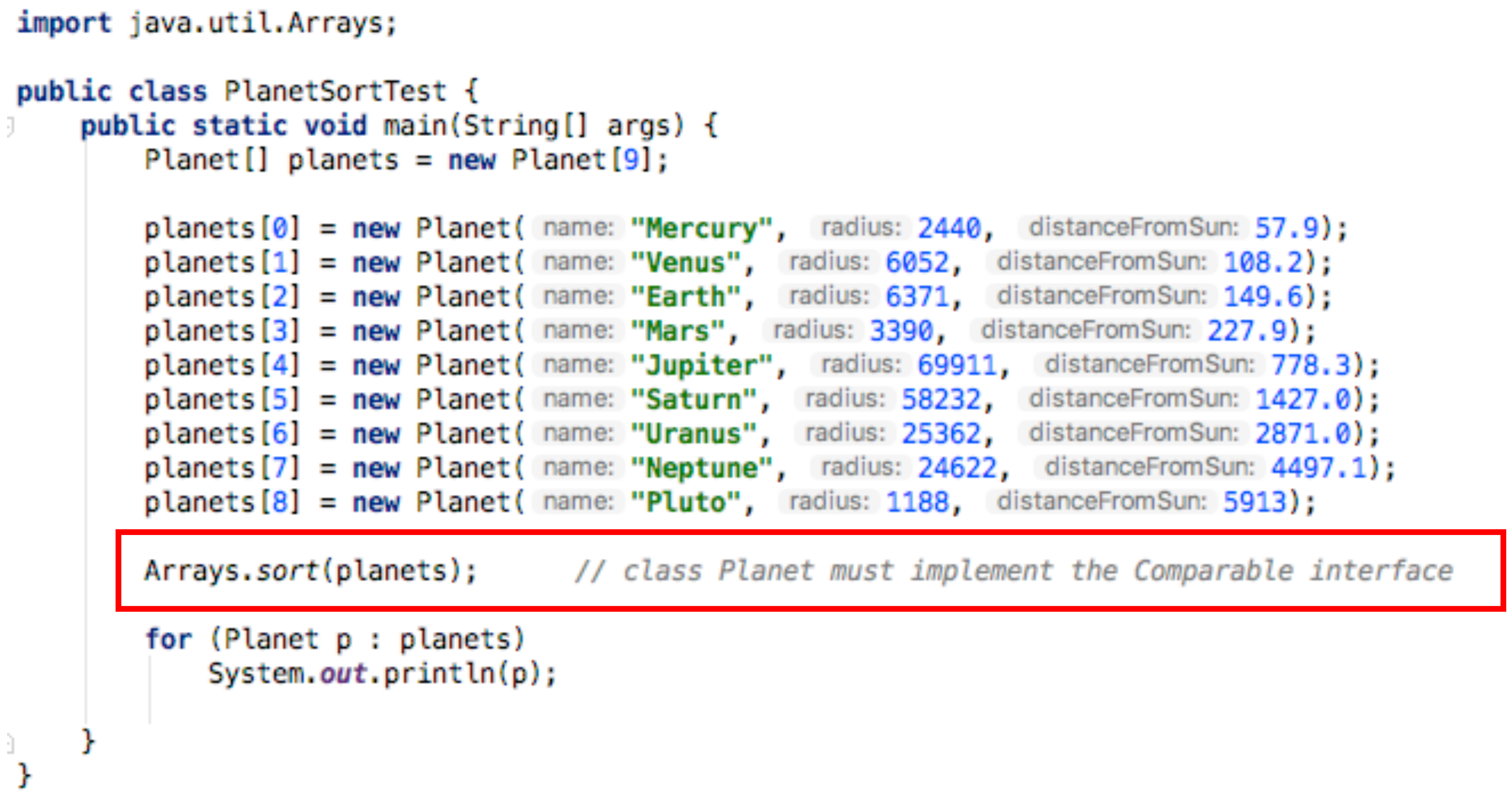
    "if the class of your array elements conforms to **Comparable** interface, then I'll sort the array for you."

    (ref: [https://docs.oracle.com/javase/7/docs/api/java/util/Arrays.html#sort(object[])](https://docs.oracle.com/javase/7/docs/api/java/util/Arrays.html#sort(object[])))



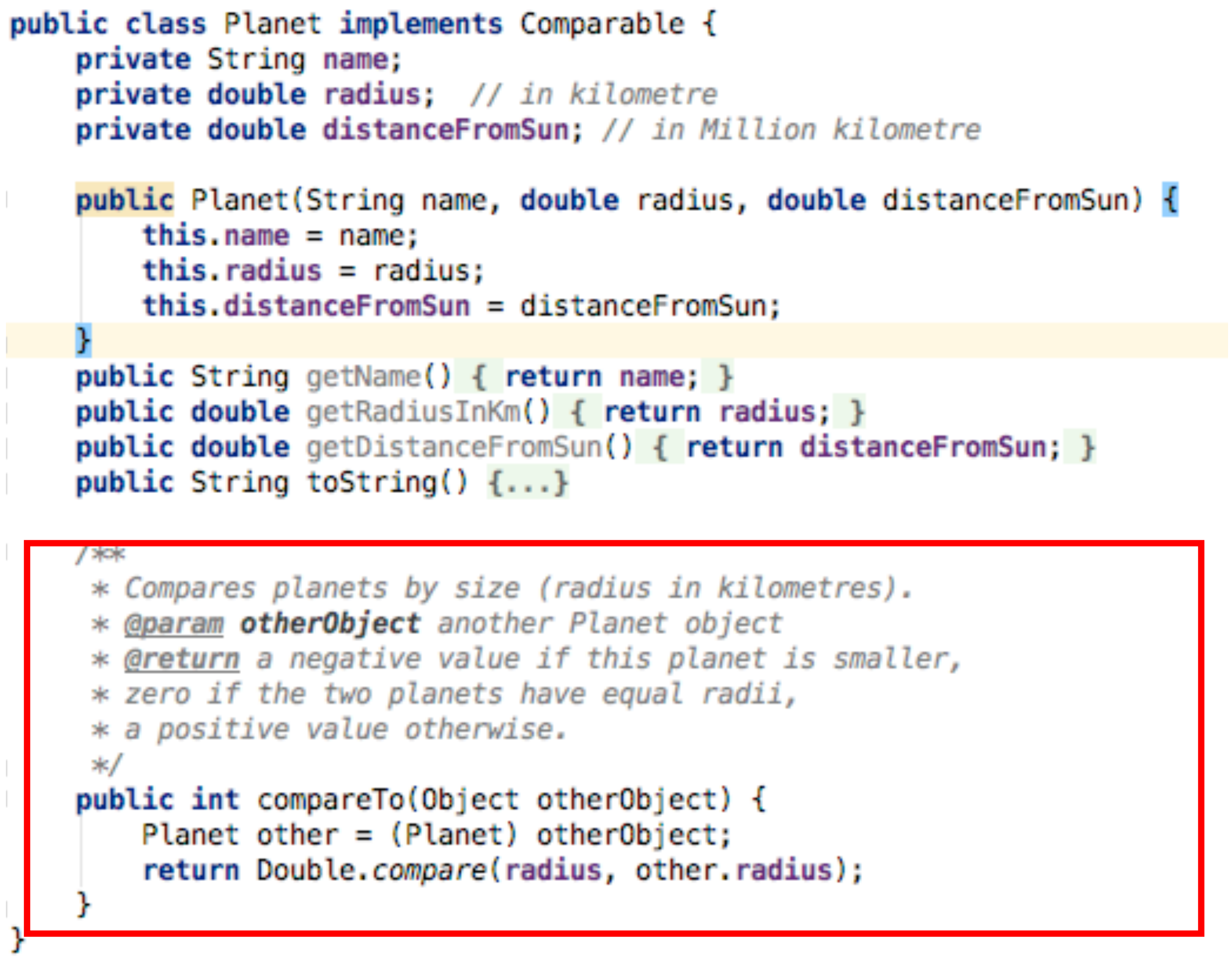
```
public interface Comparable
{
   int compareTo(Object other);  // automatically public
}
```

Example: Sorting Planets by Radius Size



```
import java.util.Arrays;

public class PlanetSortTest {
    public static void main(String[] args) {
        Planet[] planets = new Planet[9];

        planets[0] = new Planet( name: "Mercury",  radius: 2440,  distanceFromSun: 57.9);
        planets[1] = new Planet( name: "Venus",  radius: 6052,  distanceFromSun: 108.2);
        planets[2] = new Planet( name: "Earth",  radius: 6371,  distanceFromSun: 149.6);
        planets[3] = new Planet( name: "Mars",  radius: 3390,  distanceFromSun: 227.9);
        planets[4] = new Planet( name: "Jupiter",  radius: 69911,  distanceFromSun: 778.3);
        planets[5] = new Planet( name: "Saturn",  radius: 58232,  distanceFromSun: 1427.0);
        planets[6] = new Planet( name: "Uranus",  radius: 25362,  distanceFromSun: 2871.0);
        planets[7] = new Planet( name: "Neptune",  radius: 24622,  distanceFromSun: 4497.1);
        planets[8] = new Planet( name: "Pluto",  radius: 1188,  distanceFromSun: 5913);

        Arrays.sort(planets);      // class Planet must implement the Comparable interface

        for (Planet p : planets)
            System.out.println(p);

    }
}
```

Example: Sorting Planets by Radius Size

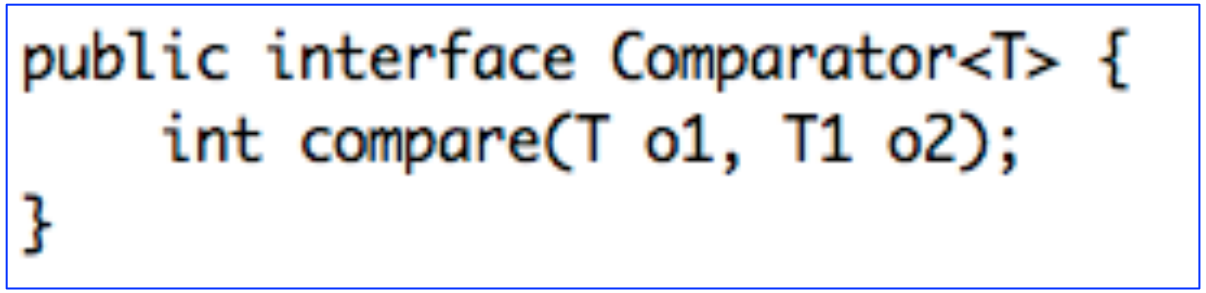


```java
public class Planet implements Comparable {
    private String name;
    private double radius;  // in kilometre
    private double distanceFromSun; // in Million kilometre

    public Planet(String name, double radius, double distanceFromSun) {
        this.name = name;
        this.radius = radius;
        this.distanceFromSun = distanceFromSun;
    }
    public String getName() { return name; }
    public double getRadiusInKm() { return radius; }
    public double getDistanceFromSun() { return distanceFromSun; }
    public String toString() {...}

    /**
     * Compares planets by size (radius in kilometres).
     * @param otherObject another Planet object
     * @return a negative value if this planet is smaller,
     * zero if the two planets have equal radii,
     * a positive value otherwise.
     */
    public int compareTo(Object otherObject) {
        Planet other = (Planet) otherObject;
        return Double.compare(radius, other.radius);
    }
}
```

What if you want to sort planets by distance?

- <u>Option 1</u>: re-implement the **compareTo** method of the Planet class
- <u>Option 2</u>: use the **Comparator** version of Arrays.sort.

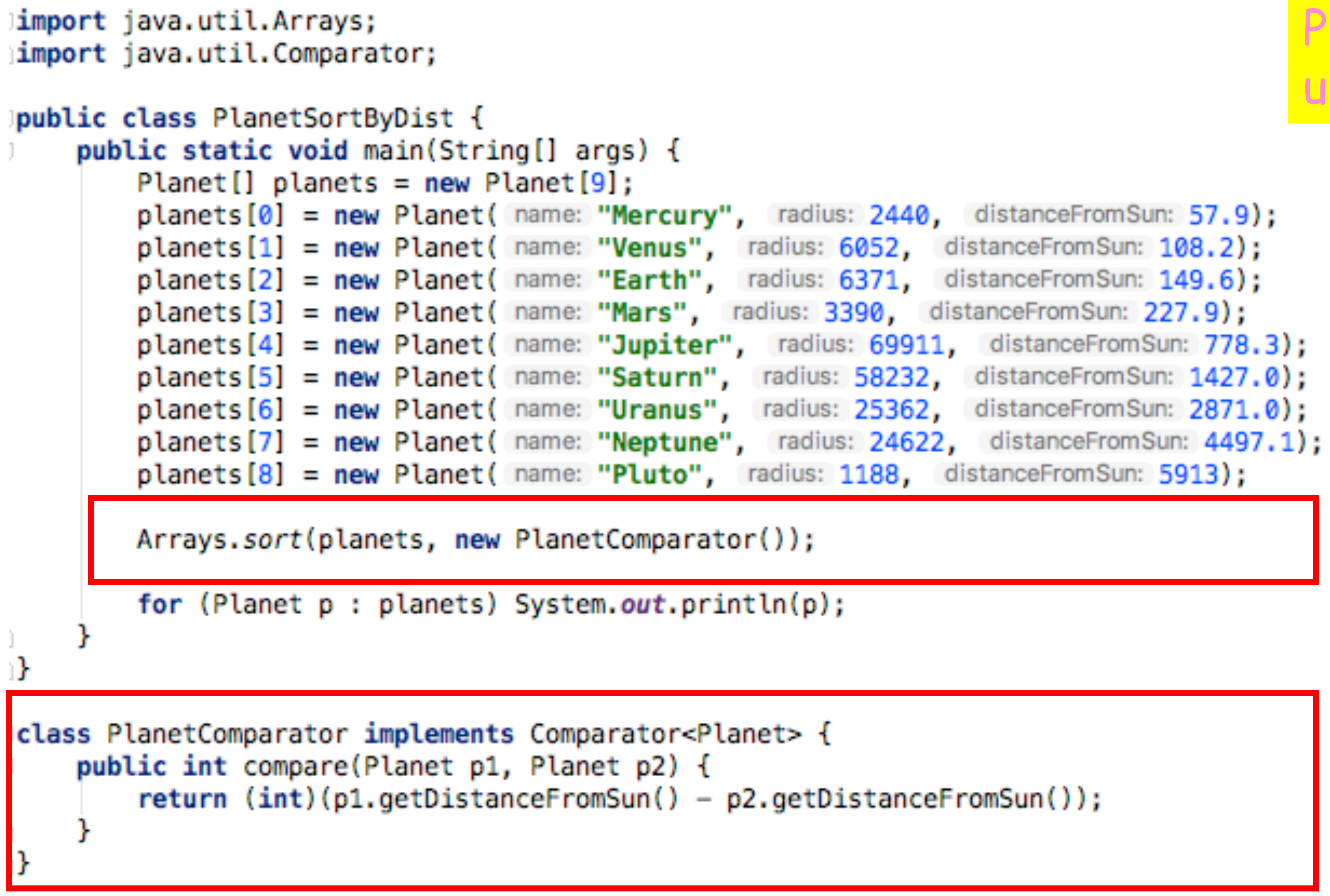
  (ref: https://docs.oracle.com/javase/7/docs/api/java/util/Comparator.html)



```
public interface Comparator<T> {
    int compare(T o1, T1 o2);
}
```

Example: Sorting Planet by Distance from the Sun

Planet class remains unchanged!



```java
import java.util.Arrays;
import java.util.Comparator;

public class PlanetSortByDist {
    public static void main(String[] args) {
        Planet[] planets = new Planet[9];
        planets[0] = new Planet( name: "Mercury",  radius: 2440,  distanceFromSun: 57.9);
        planets[1] = new Planet( name: "Venus",  radius: 6052,  distanceFromSun: 108.2);
        planets[2] = new Planet( name: "Earth",  radius: 6371,  distanceFromSun: 149.6);
        planets[3] = new Planet( name: "Mars",  radius: 3390,  distanceFromSun: 227.9);
        planets[4] = new Planet( name: "Jupiter",  radius: 69911,  distanceFromSun: 778.3);
        planets[5] = new Planet( name: "Saturn",  radius: 58232,  distanceFromSun: 1427.0);
        planets[6] = new Planet( name: "Uranus",  radius: 25362,  distanceFromSun: 2871.0);
        planets[7] = new Planet( name: "Neptune",  radius: 24622,  distanceFromSun: 4497.1);
        planets[8] = new Planet( name: "Pluto",  radius: 1188,  distanceFromSun: 5913);

        Arrays.sort(planets, new PlanetComparator());

        for (Planet p : planets) System.out.println(p);
    }
}

class PlanetComparator implements Comparator<Planet> {
    public int compare(Planet p1, Planet p2) {
        return (int)(p1.getDistanceFromSun() - p2.getDistanceFromSun());
    }
}
```

Key Properties of Java Interfaces

- Since an interface is not a class, you cannot instantiate it

```
x = new Comparable( … ); // Error
```

- You can declare variables of interface type:

```
Comparable x;   // OK
```

- The variable of interface type must refer to an object of a class that implements the interface:

```
x = new Planet(…);   // OK: because Planet class implements Comparable
```